Arquitectura de Computadores:

Tema 1:
Entrada/Salida
(cont. de LEC)

Bibliografía recomendada

- Stallings, W. “Organización y arquitectura de computadores”. Prentice Hall. 7ª Edición. 2006.
- Patterson, D. A., Hennessy, J. L. “Computer Organization and Design”. Morgan-Kaufmann. 4ª edición. 2009.
- de Miguel, P. “Fundamentos de los computadores”. Paraninfo. 9ª edición. 2004

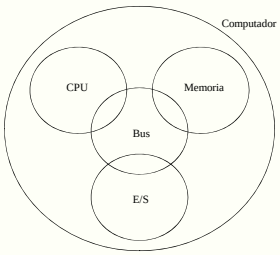
Índice

Índice

1. Introducción a la E/S y periféricos
   1. Introducción: diversidad, uso, clasificación
   2. Características: V_{transf}, t_{acc}, formato, etc
2. Módulos de E/S
3. Instrucciones de E/S
   1. Tipos
   2. Direccionamiento
   3. Ciclos de bus
4. Técnicas de E/S
   1. E/S programada
      1. Introducción
      2. Ejemplo: módulo de E/S + *driver*
      3. Visión cuantitativa

Periféricos

1. Introducción



Periféricos

- Ejemplos:
  - "Domésticos":
    - Ratón
    - Teclado
    - Impresora
    - Disco duro (HDD)
    - LANCE (Local Area Network Controller for Ethernet)
    - Etc., etc.
  - "Industriales"
    - Sensor de temperatura
    - Motores para la orientación de un telescopio
    - Sistema de control de actitud de un satélite artificial
    - Etc., etc.

Periféricos

- Gran diversidad:
  - Modo de funcionamiento
  - Formato y tamaño de los datos
  - Velocidad de transferencia
  - Tiempo de acceso
- Una posible clasificación:
  - Almacenamiento ➔ *Parallel I/O*
  - Comunicación:
    - Humanos ➔ Multimedia, *Brain interfaces*
    - Computadores ➔ Redes: *cluster, Grid computing*
    - "Medio físico" ➔ Sist. de control o *embedded systems*

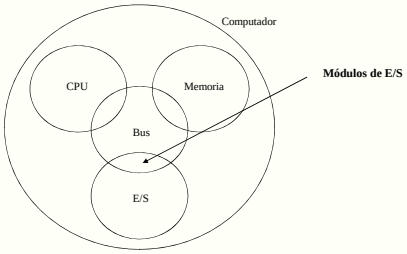
Problemática de la E/S

Gran diversidad de periféricos con características muy diferentes

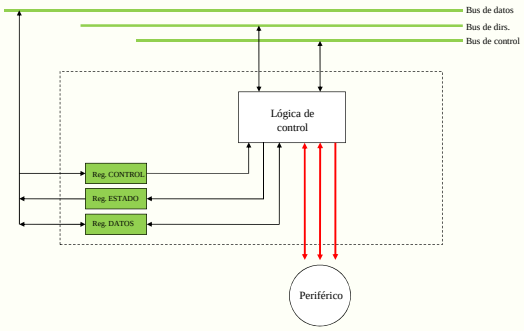
➔ es necesario "unificar" la visión Hw de los periféricos

➔***Módulos de E/S***

Módulos de E/S



Módulos de E/S



Instrucciones de E/S

- Instrucciones de la Arquitectura del computador para la transferencia entre los regs. de la CPU y los regs. de los Módulos de E/S:

```
OUT   .R1, /Dir_Reg1_Px; (orig → dest)
IN    .R1, /Dir_Reg2_Px;  (dest ← orig)
```


Direcciones de E/S

- Mapa de direcciones único: mismo ciclo de bus
  - Única línea de control MEMRQ (Memory Request) o AS (Address Strobe).
  - Las instrucciones de E/S son las mismas que las de M, **ld** y **st,** y se distinguen por la dirección.
- Mapas separados: nuevo ciclo de bus para acceder a direcciones I/O
  - Dos líneas de control: MEMRQ e IORQ (Input/Output Request).
  - Las instrucciones de E/S son **in** y **out** y se distinguen por su código de operación.

Técnicas de E/S

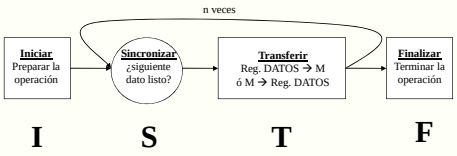
- **Técnicas de E/S**: grado de participación de la CPU en las operaciones de E/S

E/S programada
E/S por interrupciones
E/S por DMA

+
-

Técnicas de E/S

- **Operación de E/S**: transf. de un bloque de *n* palabras

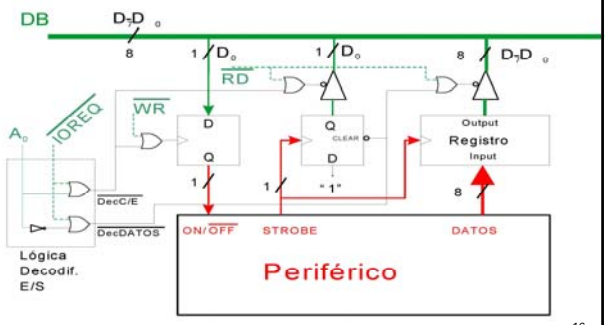


Técnicas de E/S

Técnica \ Fase	I	S	T	F
Programada	X	X	X	X
Interrupciones	X		X	X
DMA	X			X

E/S programada o directa

Módulo de E/S + *driver* para E/S programada



E/S programada o directa

Parámetros

- Tamaño del bloque: *n* palabras
- Dir. de almacenamiento en memoria: *dir_alm_M*

Direcciones de E/S

- Reg. de Control/Reg. de Estado: *Dir_C/E*
- Reg. de Datos: *Dir_Datos*

Mandatos (o 'comandos' [sic])

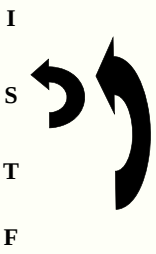
- Activar: *ON* [xxxxxxx1]
- Desactivar: *OFF* [xxxxxxx0]

Estado

- Nuevo dato listo: *LISTO* [00000001]

E/S programada o directa

```
      LD .R1,#dir_alm_M
      LD .R2,#n
      LD .R0,#H'01; ON
      OUT .R0, /Dir_C/E
sig:  IN .R0. /Dir_C/E
      CMP .R0, #LISTO
      BZ $sig
      IN .R0,/Dir_Datos
      ST .R0,[.R1++]
      DEC .R2
      BNZ $sig
      LD .R0,#H'00; OFF
      OUT .R0, /Dir_C/E
```



E/S programada o directa

Visión cuantitativa

t_{op} …
t_{CPU}…

¡¡¡MUY IMPORTANTE!!!

Índice

- E/S por **interrupciones**
  - Interrupciones
    - Interrupciones, excepciones y subrutinas
    - **Solicitud** de interrupciones
    - **Servicio** a interrupciones
      - Modo supervisor
      - Instrucciones para habilitar / inhibir el servicio de interrupciones
    - **Secuencia de reconocimiento de interrupciones (SRI)**
    - **Rutina de servicio de interrupciones**
      - Paso/actualiz. de parámetros
      - Salvar/reponer estado

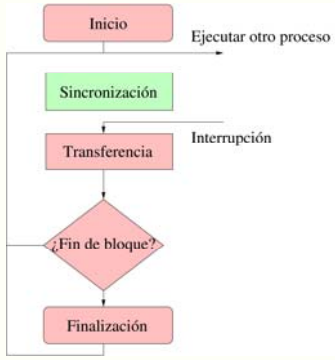
Índice

- Ejemplo de E/S por interrupciones
- Visión cuantitativa
- Operación con múltiples periféricos
  - Operación por **muestreo** *(polling)*
  - Operación mediante **vectorización**
    - Módulo de E/S para vectorización
    - Secuencia de reconocimiento de interrupciones
    - Esquema de prioridades hardware
      - Gestión centralizada
      - Gestión encadenada o *daisy chain*
    - Anidamiento de rutinas de servicio
      - Ejemplo

Índice

- Asignación de prioridades de interrupción
  - Interrupciones no enmascarables
- Conclusiones
- E/S por **acceso directo a memoria *(DMA)***
  - Robo de ciclo aislado
  - Robo de ciclo en ráfagas
  - Módulo de entrada/salida con *DMA*

E/S por interrupciones



La CPU no se encarga de la sincronización

El módulo avisa a la CPU cuando está listo para una nueva transferencia

Se ahorra mucho tiempo de CPU

Interrupciones

- Suceso asíncrono que hace que la CPU no ejecute la instrucción a la que "apunta" el PC
- Pasa a ejecutar la llamada **rutina de servicio** de **interrupción** o, simplemente, **rutina de interrupción, RTI**
- Después, la CPU debe continuar la ejecución del programa interrumpido
- Este mecanismo no puede afectar al comportamiento lógico de los programas cuya ejecución se interrumpe: rutinas de interrupción "pulcras"**¡¡¡MUY IMPORTANTE!!!**

Interrupciones, excepciones y subrutinas

Suceso	Origen	Activación	Tratamiento
Subrutinas	Interno	Síncrona	Continuar
Excepciones	Interno	Asíncrona	Cancelar
Interrupciones	Externo	Asíncrona	Continuar

- El origen es el módulo de E/S que tiene una temporización propia: *"va a su bola"*
- Pueden suceder en cualquier instante de la ejecución de una instrucción

Solicitud de interrupciones

- Mediante una nueva señal a la unidad de control: p.ej, **`INT`**
- La unidad de control (UC) secuencia un conjunto de operaciones elementales para servir o tratar la petición de interrupciones: ***SRI***

Servicio a peticiones de interrupción

- El servicio a interrupciones supone abandonar la ejecución del programa en curso y ejecutar otro programa que dé servicio a la solicitud del módulo de E/S
- Posteriormente se ha de poder continuar con el programa interrumpido
- ¿Cuándo es posible hacer esto con el menor coste posible? Al finalizar (*) la ejecución de la instrucción en curso ya que la CPU tendrá un estado consistente.

Biestable de máscara de interrupciones

- Hay dos programas ejecutando concurrentemente: el programa interrumpido y el que da servicio al módulo de E/S → condiciones de carrera
- No se puede impedir que el módulo pida interrupciones, pero sí que la UC las "vea"

Instrucciones para habilitar e inhibir la atención a interrupciones

- Existen instrucciones para inhibir **DI** ***(Disable Interrupts)*** y habilitar **EI** (***Enable Interrupts***) la atención a las interrupciones
- BMI pertenece a la parte privilegiado del **registro de estado** (RE o SR)
- Ejemplo familia x86:
  - **`CLI`** (Clear Interrupt Enable Flag) y **`STI`** (Set Interrupt Enable Flag)
  - IF (Interrupt Flag) es el bit 9 del Flags Register

Secuencia de reconocimiento de Interrupciones (SRI)

- La realiza la UC al final (*) de cada instrucción comprueba si la línea de petición de interrupción está activa
- Ha de salvar el estado necesario para posteriormente reanudar el programa interrumpido
- La siguiente instrucción que se ejecute será la primera de la rutina de servicio de interrupciones
- Se hace antes de la secuencia de *fetch*

Secuencia de reconocimiento de Int. (SRI)

```
FETCH: Si INT ∧ RE.BMI entonces:
          Salvar PC
          Salvar RE
          RE.BMI ← 1 (Inhibir interrupciones)
          RE.S ← 1 (Cambiar a modo privilegiado)
          PC ← DRTI (Dir. de la rutina de tratamiento de int.)
          ir a fetch
       Si no
          ir a fetch
```

¡¡¡MUY, MUY IMPORTANTE!!!

- Además se "**inhibe**" (prohibe) las **atención** a las Int. (para no reconocer eternamente la misma)
- Se pasa a **modo supervisor** para poder ejecutar el S.O

Instrucción de retorno de rutina de servicio de interrupción

```
RETI: Restaurar RE
      Restaurar PC
      ir a FETCH
```

- Esta instrucción deshace lo hecho en la secuencia de reconocimiento de interrupciones
  - `IRET` (familia x86)
  - `RETT` (SPARC)

Rutina de servicio de interrupciones

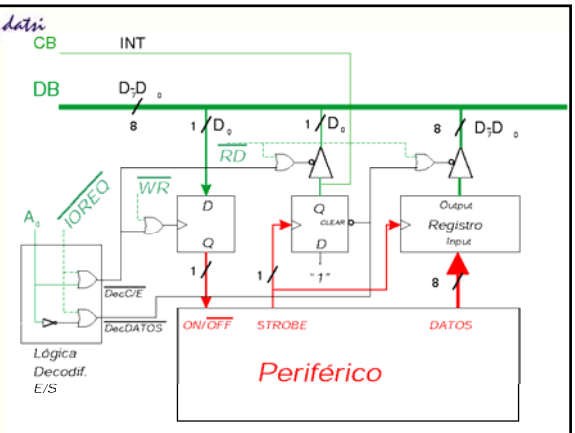
- No debe alterar la lógica del programa interrumpido: debe **salvar y restaurar** el estado que altere
- Como se ejecuta debido a un suceso asíncrono y externo no se le pueden pasar parámetros ni por pila ni en registros sino en direcciones conocidas de memoria

¡¡¡MUY, MUY IMPORTANTE!!!

- Finaliza con una instrucción `RETI`

Entrada/salida mediante interrupciones

Ejemplo con un periférico simple



Iniciación

```
LD .R1,#dir_alm_M
ST .R1,/Dir_dir_alm_M
LD .R2,#n
ST .R2,/Dir_contador
LD .R0,#H'01; ON
OUT .R0, /Dir_C/E
BR /Planificador
```

- `Dir_dir_alm_M` y `Dir_contador` contienen la dirección de memoria donde se almacenará el siguiente dato y el número de datos que quedan para completar la operación.

Rutina de servicio de interrupciones

```
PUSH .R0
PUSH .R1
PUSH .R2
LD .R1,/Dir_dir_alm_M
LD .R2,/Dir_contador
IN .R0,/Dir_Datos
ST .R0,[.R1++]
DEC .R2
CALLZ $Fin
```

```
ST .R1,/Dir_dir_alm_M
ST .R2,/Dir_contador
POP .R2
POP .R1
POP .R0
RETI

Fin:  LD .R0,#H'00; OFF
      OUT .R0, /Dir_C/E
      RET
```


E/S por interrupciones

Visión cuantitativa

t_{op} …
t_{CPU}…

¡¡¡MUY IMPORTANTE!!!